Atty Docket No. 10018010-1
(HDP Ref. No. 6215-000057)

INTERFACE TECHNOLOGY FOR MOVING DATA VIA A THIRD PARTY COPY ENGINE

[0001] **FIELD OF THE INVENTION**

[0002] The invention is generally directed to the field of data backup, and more particularly to a technology by which a host computer is unloaded from having to carry out a copy operation of data from an external source to an external destination via a third party copy engine ("3PCE"), and more particularly to an interface technology by which an application program running on a host can cause a 3PCE to carry out the desired copy operation.

[0003] **COPYRIGHT NOTICE**

[0004] A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyright rights whatsoever.

[0005] **BACKGROUND OF THE INVENTION**

[0006] The need to preserve data integrity requires that a primary storage location for data be backed-up, i.e., that a copy of the data on the primary storage location be made and kept at least one secondary storage location. Where the primary storage location is a non-volatile storage device used by a server connected to a network, the server must copy the data from its non-volatile storage device to a secondary back-up device. Given the great volume of data processed and generated each day in the present computing environment, a copy operation to backup data can significantly diminish the bandwidth of a server that is otherwise available to a client.

[0007] To combat this problem, the American National Standards Institute (ANSI) modified the small computer system interface (SCSI) standard to include an EXTENDED COPY command within its command set (see, e.g., SCSI version T10/99-143rl or T10/1236-D revision 18). The EXTENDED COPY command unloads a server from having to carry out the actual read and write operations associated with backing up data by moving that burden to a third party copy engine ("3PCE"). This is shown in more detail via the depiction of system 100 in Fig. 1.

[0008] In Fig. 1, a host/server 102 is shown as currently running three processes corresponding to application programs 104, 106 and 108. Such application programs can be data backup programs such as the OPENVIEW OMNIBACK II brand of backup program available from the HEWLETT-PACKARD COMPANY, the NETBACKUP BUSINESS SERVER brand of backup and recovery software available from the VERITAS SOFTWARE COMPANY and the NETWORKER brand of backup available from LEGATO SYSTEMS INC. Any such application program, 104, 106, 108 can free the host 102 from having to carry out the actual reads and writes associated with a backup process by using the SCSI EXTENDED COPY command.

[0009] The EXTENDED COPY command is sent to a 3PCE 112 directly or via a packet transformer 110 such as a FIBRE CHANNEL switch (which is shown as a dashed box in Fig. 1 because it is optional). An optional part of the communication path between the packet transformer 110 and the application programs 104, 106 and 108 can be the internet 120 (accordingly shown as a dashed cloud in Fig. 1). The 3PCE 112 responds to the EXTENDED COPY command by transferring data from one or more source devices to one or more target devices.

[0010] In Fig. 1, the storage devices are shown as two RAID (redundant array of independent disks) device 114 and 116 as well as a tape drive 118, for example. The RAID device 114, 116 and the tape drive 118 are each shown as communicating with the 3PCE 112 optionally through the internet 120 as well as through an optional storage area network (SAN) 124. Alternatively, one or more of the RAID device 114, 116 and the tape drive 118 could be directly connected to the third party copy engine 112. Typically, the third party copy engine 112 will either be external to the storage devices, e.g., the INTELLIGENT STORAGE ROUTER brand of router available from CHAPARRAL NETWORK STORAGE, Inc., or internal to the storage device itself, e.g., the SURESTORE brand of RAID available from the HEWLETT-PACKARD COMPANY.

[0011] SUMMARY OF THE INVENTION

[0012] The invention, in part, provides an interface technology by which an application program running on a host can invoke a command to cause a 3PCE to carry out a desired copy operation without actually generating the command needed by the 3PCE.

[0013] The invention also is, in part, directed to a first code arrangement on a computer-readable medium, execution of which causes a processor to generate a second code arrangement representing a third party copy command and its corresponding parameters. Such a first code arrangement comprises: a calling portion in response to which said processor is operable to begin said generation process; and at least one data entity portion upon which said generation process operates. Each data entity portion identifies a third party copy device ("3PCE") to carry out the copy process, a destination device to receive the copied data, desired data that is to be copied and a source of said desired data.

[0014] The invention also is, in part, directed to a liaison system interposed between an application program running on a host and a third party copy engine (3PCE) external to said host, said application program needing to copy desired data from a source device external to said host to a destination device external to said host via said 3PCE. Such a liaison system comprises: an application program interface (API) to receive the following first code arrangement from said application program, namely a calling portion, and at least one data entity portion that identifies said 3PCE, said destination, said desired data and said source; and a copy command generator to generate a second code arrangement representing a third copy command and corresponding parameters thereof that will cause said 3PCE to copy said desired data from said source to said destination; wherein said generator begins operation in response to said calling portion; and wherein said generator is operable to generate said second code arrangement based upon said data entity portion.

[0015] The invention also, in part, is directed toward a method of interfacing between application program running on a host and a third party copy engine (3PCE) external to said host, said application program needing to copy desired data from a source device external to said host to a destination device external to said host via said 3PCE. Such a method comprises: receiving a first code arrangement from said application program, said first code arrangement including a calling portion and at least one data entity portion that identifies said 3PCE, said destination, said desired data and said source; initiating, in response to said calling portion segment, a process to generate a second code arrangement representing a third party copy command and corresponding parameters thereof that will cause said 3PCE to copy said desired data from said source to said destination; and forming, once said process is begun, said second code arrangement based upon said data entity portion.

[0016] The invention also, in part, is directed toward a computer-readable medium having code portions embodied thereon that, when read by a first processor, cause a second processor to perform such a method.

[0017] Additional features and advantages of the invention will be more fully apparent from the following detailed description of the preferred embodiments, the appended claims and the accompanying drawings.

[0018] BRIEF DESCRIPTION OF THE DRAWINGS

[0019] The accompanying drawings are: intended to depict example embodiments of the invention and should not be interpreted to limit the scope thereof; and not to be considered as drawn to scale unless explicitly noted.

[0020] Fig. 1 is a block diagram depiction of a third party copy engine (3PCE) architecture according to the background art;

[0021] Fig. 2 is a depiction of a 3PCE architecture according to an embodiment of the invention;

[0022] Fig. 3 is a block diagram of some of the components within a server;

[0023] Fig. 4 is a second embodiment of a 3PCE architecture according to the invention;

[0024] Fig. 5 is a version of Fig. 4 that is more detailed in some respects as well as more simplified in some respects;

[0025] Fig. 6 is a flowchart of process steps carried out by an application program that makes use of the liaison according to an embodiment of the invention; and

[0026] Fig. 7 is a flowchart depicting process steps carried out by the liaison according to an embodiment of the invention.

[0027] DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0028] Fig. 2 is a depiction of a 3PCE architecture according to an embodiment of the invention. The system 200 of Fig. 2 includes a host/server 201 that is running application programs 206, 208 and 210. These programs correspond to programs 104, 106 and 108 of the background art except for their use of the copy agent (or liaison) 204 according to an embodiment of the invention as well as the application program interface (API) 202 (according to an embodiment of the invention).

The third party copy engine ("3PCE") architecture of Fig. 2 shows the 3PCE 112 as being physically distinct from the data storage devices 114, 116 and 118. The host 201 is shown as connecting to the 3PCE 112 through an optional packet transformer 110 (such as a FIBRE CHANNEL SWITCH) and also optionally through the internet 120.

[0029] The FIBRE CHANNEL switch 110 encapsulates SCSI commands in a protocol that can be sent over much longer distances. An example of an external 3PCE 112 is the INTELLIGENT STORAGE ROUTER brand of router available from CHAPARRAL NETWORK STORAGE, Inc., mentioned above. Alternatively, a different packet transformer 110 could be used, such as one that transforms into the ISCSI protocol (also known as SCSI/IP). Regardless, nothing about the invention is limited to the FIBRE CHANNEL environment.

[0030] The 3PCE 112 is shown as communicating with each of the storage devices 114, 116 and 118 optionally through the internet 120 as well as a storage area network (SAN) 124. As with the background art Fig. 1, optional components are illustrated in dashed lines.

[0031] Three application programs 206, 208 and 210 have been depicted in Fig. 2 for illustration purposes. The appropriate number of application programs will vary depending upon the circumstances of the server. Similarly, Fig. 2 depicts the storage devices as two RAID (redundant array of independent disks) devices 114, 116 and a tape drive 118. The actual storage devices used will depend upon the circumstances to which the copy command is applied.

[0032] In contrast, the 3PCE 112 of Fig. 2 is external to each of the backup storage devices 114, 116 and 118. The 3PCE 404 communicates with the other storage devices 116 and 118 through the optional SAN 124. An example of such a RAID device 402 is the type of RAID device made available by the HEWLETT-PACKARD COMPANY, mentioned above.

[0033] Fig. 5 is a version of Fig. 4 that is more detailed in some respects as well as more simplified in some respects (e.g., the packet transformer 110, the internet 120 and the tape drive 118 are not shown). The architecture 500 depicts an application program 50 that includes a resolve agent 504. The application program 506 is similar to any one of the application programs 206, 208 or 210, except that the resolve agent 504 is depicted as a discrete component, whereas it is considered integral (to the extent that it is present) in the application programs 206, 208, and 210. In addition, the host/server 502 includes an operating system (OS) layer 508, a file system 510 and LVM/device

512 that together correspond to the TCP/IP stack, and one or more device drivers 514 that together correspond to the network interface capability.

[0034] In Fig. 5, the resolve agent 504 communicates with each of the storage devices 402 and 118 as well as the 3PCE 404 directly via the communication paths 518, 520 and 516, respectively. However, each of those communication paths travel through the file system 510, the LVM/device 512 and the device driver(s) 514. The copy agent 204 is depicted as communicating directly with the 3PCE 404 via communication path 522. However, that communication between the copy agent 204 and the 3PCE 404 goes through the device driver(s) 514. Again, desired data on the RAID array 402 is moved to the tape drive 118 via the 3PCE 404 without the desired data passing through the host 502, as indicated by the paths 524 and 526.

[0035] The operation of the liaison or copy agent 204 according to an embodiment of the invention will now be explained. Any one of the application programs 206, 208 and 210 can call the copy agent 204 by using an appropriate request (also referred to as a first code arrangement) that conforms to the protocol of the API 202. An example of such a protocol follows. The first code arrangement can take the form of source code, or machine-executable code, etc.. The copy agent 204 generates one or more instances of a second code arrangement representing the actual third party copy command and its associated parameters in response to one request (or first code arrangement) from an application program 206/208/210.

[0036] This example has been written in the C language, but other high level languages could be used. The example is intended to run under the HPUX 11.0 brand of UNIX OPERATING SYSTEM made available by the HEWLETT-PACKARD COMPANY. Though couched in terms of the HPUX 11.0 Operating System, the example can be adapted to other operating systems such as WINDOWS, or LINUX or other versions of UNIX,. The copy agent 202 runs at the application level.

[0037] The example is couched at the source code level, so that the first code arrangement can take the form of the following alphanumeric string:

[0038] int **gen_ex_copy**(data_entity_t de[ ], int de_ent, u_char ILD[ ], int ILD_ent, u_char rcr[ ])

[0039] Here, the calling portion of the code arrangement is "gen_ex_copy" and the data entity portion, or inputs to the copy agent 202, are the parameters within the parentheses. At the machine-executable code level, the calling portion and the data entity portion are read from memory into register space of a processor in the server, i.e., are "received" by the processor, and then the processor links to the machine-executable code representing the copy agent.

[0040] The input **de**[ ] is an array of "data entities". Each element of the array describes the data to be moved, the associated source(s), destination, and 3rd party copy device. The input de[ ] is defined below as a "data_entity_t" type of data structure. The definition of the data_entity_t data structure itself includes the definitions of three other data structures, the "dev_desc" type, the "data_src_t" type and the "data_desc_t" type.

[0041] The input **de_ent** is the number of valid "de" array entries.

[0042] The input **ILD**[ ] is a character array of inline data to be included in the "data to be moved". Inline data is provided for the purposes of annotating the data that is to be backed-up. This enhances data recovery because the inline data provides context for the restoration.

[0043] The input ILD_ent is the number of valid ILD array entries.

[0044] The input **rcr**[ ] is an array used to provide Receive Copy Results (another SCSI command) data to the calling program (e.g., 206) in the event of an unsuccessful completion. The first two bytes should contain the length of the array so that the gen_ex_copy function knows the limit of the Receive Copy Results data to be included in the array as output.

[0045] A single "data entity" might result in the generation of multiple EXTENDED COPY commands in the situation in which the amount of data to be copied is more than the 3PCE can accommodate in one copy operation. This can be necessary because operational parameters of the 3PCE limit the amount of data which can be processed in a single EXTENDED COPY command.

[0046] The structure of the data entity, mentioned above, and its subordinate structures will now be explained.

[0047] The data_entity_t type of data structure is defined as follows.

```
[0048]      typedef struct data_entity
[0049]      {
[0050]          char        dev_fn[80];    // 3rd Party Copy Engine Descriptor
[0051]                                     // This is a character string used to
[0052]                              // Open and Reference the 3rd Party
[0053]                              // Copy Device

[0054]      dev_desc_tdes_dev;      // Description of the Destination Device
[0055]      int        nae;         // # of Entries in the following data_src_t array
[0056]                              //Currently a Maximum of 20 is// supported

[0057]      data_src_t ts[MAX_TS];  // Description of Source Devices and Data to
[0058]                              // Move.  This is referred to as the
[0059]                              // Target-Segment Array
[0060]    } data_entity_t;
```

[0061] The dev_desc_t type of data structure, which is used above by the data_entity_t data structure, describes the destination device. It is defined as follows.

```
[0062]    typedef struct dev_desc
[0063]    {
[0064]                              //
[0065]      u_char     d_wwn[8];    // Device WWN
[0066]      u_char     d_lun[8];    // Device LUN
[0067]      u_char     d_nport[3];  // Device N-Port
[0068]      u_long     d_bl;        // Device Block Length
[0069]      u_char     d_tid        // SCSI Target Identifier (Descriptor Type E3)
[0070]      u_char     d_type;      // Peripheral Device Type (See SCSI Spec)
[0071]      u_int      d_flags;     // Control Flags, See Description Below
[0072]                              // Bit 31 Source Device Description
[0073]                              // Bit 30 Destination Device Description
[0074]                                  // Bit 29 In-Line Device Description
[0075]                                  // Bit 28 Reserved
[0076]                                  // Bit 27 PAD Bit
[0077]                                  // Bit 26 Reserved
[0078]                                  // Bit 25 WWN Field Valid
[0079]                                  // Bit 24 LUN Field Valid
[0080]                                  // Bit 23 N-Port Field Valid
[0081]                                  // Bit 22 Target-ID Field Valid
[0082]                                  // Bit 21 Reserved
[0083]                                  // Bit 20 Fixed Bit
[0084]                                  // Bit 19 – Bit 00 - Reserved
[0085]          } dev_desc_t;
```

[0086] The data_src_t type of data structure, which is used above by the data_entity_t data structure, describes the source device(s). It is defined as follows.

```
[0087]          typedef  struct  data_src
[0088]          {
[0089]              dev_desc_t    src_device;        // Description of a Source Device
[0090]              u_int         n_a_e;             // Number of Array Entries
[0091]              data_desc_t   d2m[MAX_D2M];
[0092]          } data_src_t;
```

[0093] The data_src_t type of data structure, which is used above by the data src_t data structure, describes the data to be copied. It is defined as follows.

```
[0094]   typedef struct data_desc
[0095]   {
[0096]       u_char   sdlba[8];     // Logical Block Address of the Data to Move
[0097]                              // For ILD this is the index into the ILD
[0098]       char     ddlba[8]      // Logical Block Addr to Place Blk to Blk
[0099]                              // Data Moved
[00100]      int      sdlt;         // Stream Device Transfer Length
[00101]      u_char   nblks[8];     // Num of Logical Blocks to Move
[00102]                             // For ILD this is the number of bytes to
[00103]                             // Move
[00104]      int      flags;        // Control Flags, See Description Below
[00105]                             // Bit 31 – Bit 01 – Reserved
[00106]                             // Bit 00 CAT Bit
[00107]  } data_desc_t;
```

[00108] The reserved bits listed in the dev_desc_t type of data structure and the data_src_t type of data structure do not always have to be reserved. Rather they can be assigned as circumstances dictate, e.g., for extensibility.

[00109] This function, i.e., the copy agent 202, returns an integer. A zero return indicates successful completion of the data move. Below are the failure code listings. This list is not exhaustive, for simplicity. Only the low order 3 bytes are relevant. If there are bits set in the upper byte, they should be ignored.

```
[00110]  //  0x00000000  Successful Return
[00111]  //
[00112]  //  0x00000001  Error during Inquiry Request to 3rd Party Copy Device
[00113]  //
[00114]  //  0x00000002  Non Supported 3rd Party Copy Device
[00115]  //
[00116]  //  0x00000003  Number of "Data Entities" <= 0
```

[00117] //

[00118] // 0x00000004 Number of InLine Data Bytes < 0

[00119] //

[00120] // 0x00000005 Open Error for 3rd Party Copy Device

[00121] //

[00122] // 0x00000006 SCSI Command Error

[00123] //

[00124] // 0x00000007 Parameter Length List Error

[00125] //

[00126] // 0x00000009 Destination Device Not a Stream Device

[00127] //

[00128] // 0x0000000a In Line Data Segment Not Supported

[00129] //

[00130] // 0x0000000b Embedded InLine Data Exceeds Limit

[00131] //

[00132] // 0x00001000 Operational Parameter Error

[00133] //

[00134] // 0x00001002 Target Descriptor Build Error

[00135] In the example above, each element in the data_entity_t array will cause one or more SCSI EXTENDED COPY commands to be generated. Also, each element within the array data_entity_t can specify only one destination device and only one 3PCE. One of ordinary skill in the art will recognize that this data structure permits different elements in the array to specify different destination devices and/or 3PCEs.

[00136] Despite having limited each SCSI EXTENDED COPY command to only one destination, multiple SCSI EXTENDED COPY commands can be requested by an application program so as to, in effect, copy data to multiple destinations.

[00137] The example of the copy agent 204 given above, namely the function "gen_ex_copy" will support version T10/99-143rl or T10/1236-D revision 18) of the EXTENDED COPY command.

However, one of ordinary skill in the art would understand that the disclosed example can be readily modified to support other versions.

[00138] The function or copy agent 202, again, is executed by the server 201. The steps performed by the copy agent are depicted in Fig. 7.

[00139] In Fig. 7, flow starts at step 702 and progresses to step 704, where the copy agent 204 accepts and verifies the input parameters that were received via the API 202 from the application program, e.g., 206, for each element in the data element array, e.g., de[ ] in the example above. Again, those input parameters correspond to an identification of the 3PCE that is to carry out the copy process, an identification of a destination device to receive the copied data, an identification of the desired data itself that is to be copied and an identification of one or more source devices having the desired data.

[00140] At step 706, the copy agent 204 obtains inquiry data from the 3PCE 112/404, via the SCSI command, INQUIRY, in order to check if the copy agent 204 supports the 3PCE 112/404. Step 706 is provided to avoid the situation in which the EXTENDED COPY command is generated only to find that the 3PCE 112/404 is not supported, thus wasting computational resources of the server 201. If it is discovered at step 706 that the 3PCE 112/404 is not supported, the copy agent 204 will terminate and return an error.

[00141] At step 708, the copy agent 204 obtains the operational parameters supported by the 3PCE. These are the limitations of the 3PCE 112/404 that must be adhered to in order to generate a valid EXTENDED COPY command (and associated parameter list). At step 710, the copy agent 204 verifies that it can generate an EXTENDED COPY command that will adhere to all of the operational parameters required by the 3PCE 112/404. If not, then the copy agent 204 will terminate and return an error. For example, it might not be necessary for the copy agent 204 to use all of the operational parameters to generate an EXTENDED COPY command. So the copy agent 204 will verify that the 3PCE 112/404 can tolerate receipt of fewer than all of its operational parameters.

[00142] At step 712, the copy agent 204 checks if the amount of desired data specified by the element of the data entity array, e.g., de[ ], exceeds the single copy/move capacity of the 3PCE 112/404. If so, multiple EXTENDED COPY commands will have to be generated to move the total amount of desired data.

[00143] At step 714, the copy agent 204 builds the list of parameters for the EXTENDED COPY command. The command parameter list that is generated by the copy agent at step 714 includes target descriptors, segment descriptors, in-line data and control information. Target descriptors and segment descriptors are described in the SCSI specification, an example of which is provided above. There are a number of different target descriptors and segment descriptors. The type of descriptors which are built by the copy agent according to an embodiment of the invention will depend on the descriptors supported by the 3PCE and the information input by the application program 506. If a descriptor cannot be built due to insufficient information or inadequate descriptor support, the copy agent will exit with an error.

[00144] At step 716, the copy agent 204 builds the EXTENDED COPY command code and attaches the parameters list built in preceding step 714. At step 718, the copy agent 204 sends the copy command and the attached parameter list to the 3PCE 112/404.

[00145] At step 720, the copy agent 204 determines whether there has been an error reported by the 3PCE 112/404. If so, at step 722, the copy agent 204 sends the SCSI command, RECEIVE COPY RESULTS, to the 3PCE 112/404. Then flow proceeds to step 724, where the copy agent 204 informs the application 206 program (i.e.. the program that called it) of the unsuccessful status and provides the Error Info produced by the RECEIVE COPY RESULTS command. The flow proceeds to step 730 where flow ends.

[00146] If no error is found at step 720, then flow proceeds to step 726. At step 726, the copy agent 204 determines if all of the desired data specified by the currently considered element of the data entity array, e.g., de[ ], has been copied/moved.

[00147] If not, flow loops back up to step 714. But if so, flow proceeds to step 728, where the copy agent 204 determines whether all elements in the data entity array, e.g., de[ ], have been processed. If not, flow loops back up to step 706 to process the next element. But if so, flow proceeds to step 726, where the copy agent 204 informs the application program 206 of a successful status. No error information is provided in this situation because no error condition arose. And then flow ends at step 726.

[00148] How an application program, e.g., 206, makes use of the API 202 and the copy agent 204 is put into context by the flowchart of Fig. 6. The process starts at step 602 and proceeds to step 604 where the application program, e.g., 506, determines what methodology is to be used in order to make the data available for movement. Such methodologies can include the split methodology, the mirror methodology, the snapshot methodology, etc. At step 606, the application program 506 makes the data available. In other words, it performs the selected methodology.

[00149] At step 608, the application program 506 determines which 3PCE will carry out the copy command. This may include the application program 506 retrieving some or all of the parameters and limitations of the selected 3PCE.

[00150] At step 610, the application program 506 determines the destination of the data to be copied. In some instances, this has already been determined by the client of the application program 506.

[00151] At step 612, the application program 506 prepares the destination device for the receipt of the desired data. This can include, in the case of a tape drive, repositioning the tape and/or mounting different tapes. At step 614, the resolve agent 504 within the application program 506 determines the mapping (i.e., location) of the desired data. This deals with the possibility that the desired data can be broken up and stored in multiple physical locations/drives and yet considered, at a logical level, to be one unit.

[00152] At step 616, the application program 506 determines any in-line data that is associated with the desired data. Then at step 618, the application program 506 calls the copy agent 204 via use of the API 202. Again, the application program 506 executes a function corresponding to the copy agent 506 and passes parameters to the copy agent 506. The parameters identify the 3PCE (by whom the copy process is to be administered), the data source device(s) (from whom the desired data is to be copied), the desired data that is to be copied, the destination device (to whom the data is to be copied), the **rcr[ ]** array (to be filled by the copy agent 204) and (optionally) any in-line data.

[00153] After the copy agent 204 returns its status information and results, the application program 506 can perform error recovery at step 620, if that is appropriate. At step 622, the application program 506 notifies its client about the results of the data movement. At step 624, the process ends.

[00154] The communications being performed by the resolve agent 504 at step 614 are depicted in Fig. 5 as communication paths 516, 518 and 520. Again, the communications by the copy agent at step 618 are depicted in Fig. 5 as the communication path 522.

[00155] The example of the copy agent given above generates an EXTENDED COPY command according to the SCSI standard. However, the invention is not to be limited to only the SCSI EXTENDED COPY command. Rather, the invention has applicability to other types of data copy and data movement commands that involve the use of a third party copy engine, source devices and destination devices.

[00156] Similarly, use of the API and copy agent according to an embodiment of the invention has been couched in terms of servicing a backup program. But the invention has wider usefulness, namely to any application program needing to move or copy data via a 3PCE.

[00157] An embodiment of the invention is, in part, a recognition that robust generation of third party copy commands, such as SCSI EXTENDED COPY commands, requires a complicated piece of software that can accommodate a great many combinations of data source devices, third party copy engines (3PCEs) and destination storage devices.

[00158] An embodiment of the invention is also, in part, a recognition that it is burdensome to require each application program, such as the application programs 104, 106 and 108 in background Fig. 1, to independently provide such robust third party copy command (e.g., SCSI EXTENDED COPY command) generation capability. Alternatively, the invention also is, in part, a recognition that limiting the numbers of source storage devices, 3PCEs and destination storage devices that are supported by an application program can significantly detract from the application program's competitiveness in the market.

[00159] An embodiment of the invention also is, in part, a recognition that an application program can invoke robust third party copy command (e.g., SCSI EXTENDED COPY command) capability without the need for it to independently provide complex command generation software if a liaison (to generate the command) is provided between the application program and the third party copy engine.

[00160] An embodiment of the invention also is, in part, a recognition that such a liaison need only receive a request for a third party copy command (e.g., SCSI EXTENDED COPY command) from an application program that identifies the 3PCE to be used, a destination device to receive the copied data, identification of the desired data that is to be copied and an identification of the source of the desired data. Accordingly, the invention also, in part, provides an interface protocol by which application programs can communicate such information (namely what is to be copied, by-whom, from-where and to-where) to the liaison.

[00161] The invention being thus described, it will be obvious that the same may be varied in many ways. Such variations are not to be regarded as a departure from the spirit and scope of the invention, and all such modifications as would be obvious to one skilled in the art are intended to be included within the scope of the following claims.